Monte-Carlo Algorithms for the Improvement of Finite-State Stochastic Controllers: Application to Bayes-Adaptive Markov Decision Processes

Michael O. Duff
Department of Computer Science
University of Massachusetts at Amherst
`duff@cs.umass.edu`

Abstract

We consider the problem of "optimal learning" for Markov decision processes with uncertain transition probabilities. Motivated by the correspondence between these processes and partially-observable Markov decision processes, we adopt policies expressed as finite-state stochastic automata, and we propose policy improvement algorithms that utilize Monte-Carlo techniques for gradient estimation and ascent.

1 Introduction

This paper considers the problem of computing optimal policies, or "optimal learning" strategies, for Markov decision processes (MDP's) in which uncertainty in transition probabilities is expressed in terms of a prior distribution over possible parameter values.

A solution to this problem would constitute a solution to what, in reinforcement learning theory, is sometimes referred to as the "exploration versus exploitation" problem. The goal is to maximize some measure of total reward an agent operating in an uncertain environment derives over its entire duration of operation—an interval that may require the agent to balance two sometimes conflicting impulses: (1) greedy exploitation of its current world model, and (2) exploration of its world to gain information that can refine the world model and improve the agent's policy.

An optimal leaning strategy is optimal with respect to the prior distribution, which describes the agent's uncertainty about process parameters (transition probabilities). One way of thinking about the computational procedures that we propose is that they perform an offline computation of an online, adaptive machine. We may regard the process of approximating an optimal policy as "compiling" an optimal learning strategy, which can then be "loaded" into an agent. The agent can then be released into its environment, and with actions dictated by the compiled policy, the agent behaves in a manner that is optimal with respect to it's prior, which describes the distribution of environmental scenarios the agent is likely to encounter.

If one were to observe the agent in operation, its physical-state-to-action mapping would appear to be nonstationary; the agent would appear to adapt as it observes the consequences of its actions, but this adaptation is illusory in the sense that the agent simply follows its pre-compiled policy, which is a static, nonstationary stationary mapping from observation histories, or various compacted versions of observation histories, to actions.

In previous work (Duff, 2000), we adopted conjugate families of distributions for explicity representing and tracking evolving uncertainty, and we successfully applied a reinforcement learning-based, policy-improvement scheme in which function approximators generalized over parameters describing uncertainty (the "information state").

Here, we parameterize policies by general finite-state stochastic controllers. In this case, the controller does not explicity model or track evolving uncertainty distributions; it simply traverses its associated state-transition diagram in accordance with observed state transitions, generating actions en route.

This paper begins by summarizing several aspects of the correspondence between MDP's with uncertain transition probabilities and partially-observable Markov decision processes (POMDP's). Section 3 presents general finite-state stochastic controllers, derives systems of linear equations for the value function and the value function gradient with respect to controller parameters, and proposes a relatively direct Monte-Carlo algorithm for policy improvement. Section 4 proposes another Monte Carlo algorithm for gradient estimation that makes use of simulated process sample paths.

2 BAMDP's and POMDP's

We shall refer to models for decision making in which one adopts a Bayesian framework to model uncertainty in the transition probabilities associated with some underlying Markov decision process as *Bayes-adaptive Markov decision processes* (BAMDP's). Historically, researchers have considered the "finite-parameter" or "multi-matrix" case in which, for example, it is assumed that the underlying Markov decision process is drawn from some finite set of possible transition matrices that are assumed to be known, and, in this instance, one can cast the process straightforwardly as a classical partially-observable Markov decision process. Here we attempt to extend the correspondence to the case where our uncertainty is expressed more generally in terms of distributions over transition probabilities, rather than over some some finite set of known matrices—details are presented elsewhere.

For POMDP's, the unknown state of nature is the identity of the underlying physical state, $s(t)$, which can change with each time-step. The true state of nature is a member of a finite set. For BAMDP's, the unknown is the true *generalized transition matrix*, P, which is a (constant) matrix of all transition probabilities, p_{ij}^a (r_{ij}^a denotes the corresponding one-step rewards)—the true state of nature is a point in a compact subset of $\mathcal{R}^{AN}$(where A is the number of actions and N is the number of physical states).

For POMDP's, observations are generated via a distribution over observations given $(a(t), s(t+1))$. For BAMDP's, observations are state transitions, $i^a \to j$ (state i, action a, resulting in transition to state j) which are generated via p_{ij}^a, a distribution over $s(t+1)$ given $(s(t), a(t))$.

For POMDP's, the belief state, π, is a distribution (point mass-function) over possible physical states. For BAMDP's, the information state, $dH(P|\theta)$, is a distribution (density) over possible generalized transition matrices, parameterized by θ (Dirichlet distribution parameters, for example). Discrete and continuous versions of Bayes's rule prescribe how, respectively, the belief state and information state are to be revised in light of observation.

For POMDP's, the value function is convex and, for finite horizons, piecewise-linear. We can consider the case for BAMDP's by pursuing a path parallel to the inductive development for the POMDP case. Given that the value function with $t-1$ steps remaining can be defined in terms of a finite set of functions, $\left\{\alpha_{(t-1)}^k(j,P)\right\}_{k=1}^{K'}$, it can be shown that:

$$V^t(i, dH(P|\theta) = \max\{_a \underbrace{\int_P \sum_j p_{ij}^a \left(r_{ij}^a + \gamma\alpha_{(t-1)}^{k^*(\theta,i,j,a)}(j,P)\right) dH(P|\theta)}_{\alpha_{(t)}}\},$$

where

$$k^*(\theta,i,j,a) \equiv \arg\max_k \left\{\int_P \alpha_{(t-1)}^k(j,P)p_{ij}^a dH(P|\theta)\right\},$$

and where the underbraced term, evaluated at $\arg\max_a$, may be taken as a definition for an $\alpha_{(t)}$, which is a function of the physical state and the generalized transition matrix. This equation defines $V^t(i, dH(P|\theta)$ about a local neighborhood of some nominally chosen information state. We may associate the maximizing a with the newly-constructed $\alpha_{(t)}$.

The idea of characterizing the value function in terms of a finite set of elements thus generalizes from the POMDP case. The fundamental differences stem from the fact that, for BAMDP's, the information state is a continuous density—a function rather than a finite-dimensional vector—which calls for appropriate generalization of inner product (from summation to integration) and re-definition of α (from vector to function).

Can we parallel the development of Monahan's algorithm for POMDP's to the BAMDP case? We can generate all possible candidate $\alpha_{(t)}$-functions, and then consider how we might go about pruning this set of superfluous members.

For fixed i, there are A_i possible choices for a. In the sum over j, each term could select any of the K possiblilites for k^*. This implies a total of A_iK^N (or AK^N total for all i) possible $\alpha_{(t)}(i,P)$-functions. A particular candidate, α^j, is not superfluous if there exists some region of information-state space in which its inner product with the density dominates that of all the other α's (in that region of the domain, α^j defines V's envelope). In other words, we wish to check whether there exists a feasible θ such that

$$\int_P \alpha_{(t)}^k(i,P)dH(P|\theta) \leq \int_P \alpha_{(t)}^j(i,P)dH(P|\theta) \qquad \forall k.$$

This set of linear-functional inequalities generalizes the linear system of constraints for POMDP's, which are addressed using linear programming, but there appears to be no easy way to make use of this fact for the BAMDP case.

A more promising approach might start by adopting finite state controllers to define policies. For POMDP's the value function associated with a finite-state controller can be computed by solving a system of linear equations, and recent POMDP research has developed procedures for constructing improved controllers. For example, (Meuleau et al, 1999) directly search a set of restricted finite-state controllers for the globally-optimal deterministic policy or a locally-optimal stochastic policy.

This stochastic policy ascent approach would seem to be a promising direction to pursue with regard to BAMDP's. The main analytical job to be done is to derive an expression for the gradient of the value function of a BAMDP governed by a stochastic policy (parametrized by a finite-state controller) with respect to the policy parameters (i.e., action distributions and memory-state transition distributions). This issue, without the parenthetical qualifiers, has been addressed in (Duff, 2000), and in this paper is worked out in detail (where parenthetical qualifiers apply).

3 Finite-state stochastic controllers

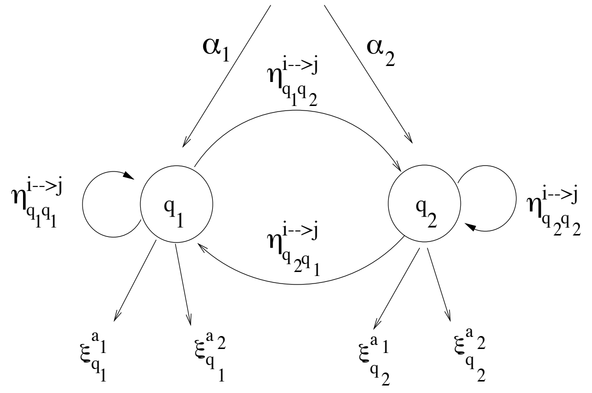
The elements of a finite state stochastic controller are:

- A finite set, Q, of memory states. (We also use Q to denote the *number* of memory states.)
- A finite set of inputs, which we take here to be the set of observable `state`$\to$`action`$\to$`next_state` triples: $i^a \to j$.
- A distribution over starting memory states: $\alpha_q \equiv Pr\{q_0 = q\}$.
- A distribution over actions for each memory state: $\xi_q^a \equiv Pr\{a|q\}$.
- A memory state transition distribution for each memory state: $\eta_{qq'}^{i^a \to j} \equiv Pr\{q'|q, i, j, a\}$.

We may think of a finite-state stochastic controller as a directed graph with action-distributions associated with vertices, which correspond to memory states, and with arcs directed to successor memory states in a way that reflects the memory-state transition distributions (see Figure 1).

Note that a policy represented by a finite-state controller explicitly maps a finite number of memory states, rather than hyperstates (as in (Duff, 2000)), to distributions over actions.

An MDP governed by a finite-state controller may be viewed as interacting automata that form a Markov chain with a state-space that is the cross product, $S \times Q$, of the underlying MDP with the finite-state controller. The (discounted) value function associated



α_1
α_2
$\eta^{i\to j}_{q_1 q_2}$
$\eta^{i\to j}_{q_1 q_1}$
q_1
q_2
$\eta^{i\to j}_{q_2 q_2}$
$\eta^{i\to j}_{q_2 q_1}$
$\xi^{a_1}_{q_1}$
$\xi^{a_2}_{q_1}$
$\xi^{a_1}_{q_2}$
$\xi^{a_2}_{q_2}$

Figure 1: A simple finite state stochastic controller with two memory states (only one representative arc for each $\eta_{qq'}^{i^a \to j}$ $\forall i, j, a$ is shown for fixed q and q').

with this hybrid process satisfies an equation expressing consistency with transitions to successor process states:

$$\begin{aligned} V((i,q)|P) &= \textstyle\sum_a \xi_q^a \sum_j p_{ij}^a \big[r_{ij}^a \\ &\quad + \gamma \textstyle\sum_{q'} \eta_{qq'}^{i^a \to j} V((j,q')|P) \big] \qquad i = 1, \ldots, N \\ &\qquad\qquad q = 1, \ldots, Q. \end{aligned} \tag{1}$$

This equation can be rewritten in the standard form, "$(I - \gamma A)x = b$," a linear system of equations in which A has dimension $NQ \times NQ$, where

$$\begin{aligned} A_{(i-1)Q+q,(j-1)Q+q'} &\equiv \textstyle\sum_a \xi_q^a p_{ij}^a r_{ij}^a \\ &\quad i, j = 1, \ldots, N \quad q, q' = 1, \ldots, Q \\ b_{(i-1)Q+q} &\equiv \textstyle\sum_j \sum_a \xi_q^a p_{ij}^a r_{ij}^a \\ &\quad i = 1, \ldots, N \quad q = 1, \ldots, Q. \end{aligned} \tag{2}$$

Given that the Markov chain begins in state i_0, the expected value of the controlled process is

$$V_{i_0} \equiv \sum_q \alpha_q V(i_0, q),$$

where, notationally, we understand that V's value is for a fixed P.

3.1 Value gradient with respect to finite-state stochastic controller parameters

We shall employ exponentialized, normalized parameterized functions to represent all controller transition probabilities. This form is differentiable with respect to its parameters and ensures that the resulting functions define valid sets of transition probabilities for the controller. For example, we parameterize the initial controller memory state using $\{\phi_q\}_{q=1}^{Q}$ via

$\alpha_q \equiv \frac{e^{\phi_q}}{\sum_{q'} e^{\phi_{q'}}}$. $q = 1, ..., Q$. Similarly, action distributions are parameterized using $\{\chi_q^a\}$ via $\xi_q^a \equiv \frac{e^{\chi_q^a}}{\sum_{a'} e^{\chi_q^{a'}}}$, $a = 1, ..., A\ \forall q$, and memory-state transtition probabilities are parameterized using $\left\{\psi_{qq'}^{i^a \to j}\right\}$ via $\eta_{qq'}^{i^a \to j} \equiv \frac{e^{\psi_{qq'}^{i^a \to j}}}{\sum_{q'} e^{\psi_{qq'}^{i^a \to j}}}$, $q' = 1, ..., Q\ \forall q, i, j, a$.

We now procede to compute the gradient of V_{i_0} with respect to all controller parameters.

First, with regard to initial memory-state parameters, it can be shown that

$$\frac{\partial V_{i_0}}{\partial \phi_{\hat{q}}} = \alpha_{\hat{q}} \left[V(i_0, \hat{q}) - \sum_q \alpha_q V(i_0, q) \right] \quad \hat{q} = 1, \ldots Q. \tag{3}$$

With regard to action parameters, starting from Equation 1, it can be shown that

$$\begin{aligned} \tfrac{\partial V(i,q)}{\partial \chi_{\hat{q}}^{\hat{a}}} \quad = \quad & \gamma \textstyle\sum_j \sum_{q'} \sum_a \xi_q^a p_{ij}^a \eta_{qq'}^{i^a \to j} \tfrac{\partial V(j,q')}{\partial \chi_{\hat{q}}^{\hat{a}}} \\ & + \delta_{q\hat{q}} \xi_{\hat{q}}^{\hat{a}} \left\{ \textstyle\sum_j p_{ij}^{\hat{a}} \left[r_{ij}^{\hat{a}} + \gamma \sum_{q'} \eta_{\hat{q}q'}^{i^{\hat{a}} \to j} V(j, q') \right] \right. \\ & \left. - \textstyle\sum_a \xi_{\hat{q}}^a \left(\sum_{j'} p_{ij'}^a \left[r_{ij'}^a + \gamma \sum_{q'} \eta_{\hat{q}q'}^{i^a \to j'} V(j', q') \right] \right) \right\}, \end{aligned}$$

where $\delta_{q\hat{q}}$ is the Kronecker delta. Finally,

$$\frac{\partial V_{i_0}}{\partial \chi_{\hat{q}}^{\hat{a}}} = \sum_q \alpha_q \frac{\partial V(i_0, q)}{\partial \chi_{\hat{q}}^{\hat{a}}}.$$

With regard to memory-state transition parameters, it can be shown that

$$\begin{aligned} \tfrac{\partial V(i,q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}} \quad = \quad & \gamma \textstyle\sum_j \sum_{q'} \sum_a \xi_q^a p_{ij}^a \eta_{qq'}^{i^a \to j} \tfrac{\partial V(j,q')}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}} \\ & + \delta_{i\hat{i}} \delta_{q\hat{q}} \gamma \xi_{\hat{q}}^{\hat{a}} p_{\hat{i}\hat{j}}^{\hat{a}} \eta_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}} \left[V(\hat{j}, \widehat{q'}) - \textstyle\sum_{q'} \eta_{\hat{q}q'}^{\hat{i}^{\hat{a}} \to \hat{j}} V(\hat{j}, q') \right], \end{aligned}$$

and

$$\frac{\partial V_{i_0}}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}} = \sum_q \alpha_q \frac{\partial V(i_0, q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}.$$

3.2 A direct Monte-Carlo policy improvement scheme

It can be seen that the formula for $\frac{\partial V(i,q)}{\partial \chi_{\hat{q}}^{\hat{a}}}$, for fixed $\hat{q}$ and $\hat{a}$, may be rewritten as a system of linear equations, "$(I - \gamma A)x = b$," where A is the same matrix that appeared in linear system for $V(i, q)$, following Equation 1, and b is a vector that is zero but for N elements corresponding to different choices of i; i.e.,

$$\begin{aligned} b_{(i-1)Q+\hat{q}} \quad = \quad & \xi_{\hat{q}}^{\hat{a}} \left\{ \textstyle\sum_j p_{ij}^{\hat{a}} \left[r_{ij}^{\hat{a}} + \gamma \sum_{q'} \eta_{qq'}^{i^{\hat{a}} \to j} V(j, q' \right] \right. \\ & \left. - \textstyle\sum_a \xi_{\hat{q}}^a \left(\sum_{j'} p_{ij'}^a \left[r_{ij'}^a + \gamma \sum_{q'} \eta_{qq'}^{i^a \to j'} V(j', q) \right] \right) \right\}, \\ & i = 1, \ldots, N. \end{aligned} \tag{4}$$

Similary, the formula for $\frac{\partial V(i,q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$, for fixed $\hat{q}$, $\widehat{q'}$, $\hat{i}$, $\hat{j}$, and $\hat{a}$, may be written in the form "$(I - \gamma A)x = b$," where A is as above and b is a vector with one nonzero element:

$$\begin{aligned} b_{(\hat{i}-1)Q+\hat{q}} \quad = \quad & \gamma \xi_{\hat{q}}^{\hat{a}} p_{\hat{i}\hat{j}}^{\hat{a}} \eta_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}} \left[V(\hat{j}, \widehat{q'} \right. \\ & \left. - \textstyle\sum_{q'} \eta_{\hat{q}q'}^{\hat{i}^{\hat{a}} \to \hat{j}} V(\hat{j}, q') \right]. \end{aligned} \tag{5}$$

To compute all the gradient components, we need only compute A^{-1} once. We then multiply A^{-1} by the appropriate b-vector to obtain the desired components. For example, to compute the gradient components with respect to initial memory state parameters, we begin by computing the value function components, $V(i_0, q)$, $q = 1, \ldots Q$, by multiplying the corresponding rows (rows $(i_0 - 1)Q + q$, $q = 1, \ldots, Q$) of A^{-1} by b_V, where b_V is the b-vector associated with the value-function linear system given previously in Equation 2. $\frac{\partial V_{i_0}}{\partial \phi_{\hat{q}}}$ is then obtained from Equation 3.

To compute the gradient components with respect to action parameters, we begin by computing the gradient components, $\frac{\partial V(i_0,q)}{\partial \chi_{\hat{q}}^{\hat{a}}}$, $q = 1, \ldots, Q$, by multiplying rows $(i_0 - 1)Q + q$, $q = 1, \ldots, Q$, of A^{-1} by $b_{\chi_{\hat{q}}^{\hat{a}}}$, where $b_{\chi_{\hat{q}}^{\hat{a}}}$ is the b-vector associated with the action parameter gradient linear system given previously in Equation 4. Since $b_{\chi_{\hat{q}}^{\hat{a}}}$ contains only N non-zero elements, only columns $(i-1)Q + \hat{q}$, $i = 1, \ldots, N$, of A^{-1} contribute. Finally, we obtain $\frac{\partial V_{i_0}}{\partial \chi_{\hat{q}}^{\hat{a}}}$ as the α_q-weighted sum of $\frac{\partial V(i_0,q)}{\partial \chi_{\hat{q}}^{\hat{a}}}$ components.

Similary, to compute the gradient components with respect to memory-state transition parameters, we begin by computing the gradient components, $\frac{\partial V(i_0,q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$, $q = 1, \ldots, Q$, by multiplying rows $(i_0 - 1)Q + q$, $q = 1, \ldots, Q$, of A^{-1} by $b_{\psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$, where $b_{\psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$ is the b-vector associated with the memory-state transition parameter gradient linear system given previously in Equation 5. Since $b_{\psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$ contains only one non-zero element, only column $(\hat{i} - 1)Q + \hat{q}$, of A^{-1} contributes. Finally, we obtain $\frac{\partial V_{i_0}}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$ as the α_q-weighted sum of $\frac{\partial V(i_0,q)}{\partial \psi_{\hat{q}\widehat{q'}}^{\hat{i}^{\hat{a}} \to \hat{j}}}$ components.

The preceding development has shown how we can compute the gradient of performance with respect to all controller parameters, given a particular value for the generalized transition matrix, P. Our ultimate goal is to compute a controller that is optimal with respect to the prior distribution over P, and a simple Monte-Carlo scheme repeatedly: (1) samples from this prior, (2) computes the value function and its gradient with respect to controller parameters, and (3) takes a small step in parameter space in the direction suggested by the exact gradient computed for the sample P.

This approach requires $O(N^2Q^2A)$ space to store the controller parameters. Inverting the A matrix exactly, using LU-decomposition for example, is an $O(N^3Q^3)$ proposition, and is required each time we sample P. Alternatively, we could apply iterative matrix-inversion techniques, at the cost of $O(N^2Q^2)$ operations per iteration, to compute an approximate inverse. Given A^{-1}, computing the value function requires $O(NQ^2)$ multiplications per iteration. Then computing the gradients with respect to *all* initial memory-state, action, and memory-state transtion parameters requires $O(Q^2)$, $O(N^2Q^2)$, and $O(N^2Q^3A)$ multiplications, respectively. It is difficult to be precise regarding the time complexity of this algorithm. For a fixed policy, the squared-error of Monte-Carlo estimates is inversely proportional to number of samples (for gradient estimates, we note that squared-error is proportional to the variance of the gradient, which may be significant).

4 Monte-Carlo gradient estimation

We begin by noting that the matrix A may be interpreted as the "policy-averaged" transition matrix associated with the hybrid MDP defined over $S \times Q$, and a Monte-Carlo approach for estimating the value function and its gradient with respect to controller parameters can make use of this interpretation. The linear systems presented in the previous section each have the form $(I - \gamma A)x = b$. Rewriting slightly, we have $x = (I - \gamma A)^{-1}b = \sum_{k=0}^{\infty}(\gamma A)^k b$.

The (i_0, q_0)th row of $\sum_k (\gamma A)^k$ may be interpreted as the expected (discounted) number of visits to hybrid process states given that we start the process in state (i_0, q_0). For an episodic, undicounted ($\gamma = 1$) problem, we can obtain an unbiased estimate for $x_{(i_0,q_0)}$ by starting the process in hybrid state (i_0, q_0), then following a simulated hybrid process trajectory and accumlating the corresponding components of b; i.e.,

$$x_{(i_0,q_0)} \approx \sum_{(i,q) \in trajectory} b_{(i,q)}.$$

For discounted problems, we terminate each step of the simulated trajectory with probability $1-\gamma$. Since the A matrix is the same for all of the linear systems, we can use the same simulated trajectory to estimate the value function and all of its gradients as well. The foregoing interpretation suggests a Monte-Carlo algorithm of the following form:

- Initialize all controller parameters.
- Do forever:
  - Sample the prior distribution to obtain a generalized transition matrix, P.
    - ∗ Set the initial physical state to i_0, and sample the initial-memory state distribution, α, to obtain q_0.
    - ∗ While the trajectory has not been terminated,
      - · Call the current hybrid state (i, q).
      - · Sample ξ, P, and η to obtain action a, next-state i', and next-memory state q' (and reward $r^a_{ii'}$).
      - · Update the value function estimate; for instance, by performing a TD(0) update.
      - · Update the estimates of value gradient; various components of the b-vectors specified in Section 3.2 determine incremental contributions.
      - · Update the controller parameters by moving in the direction of the gradient.
      - · Let $(i, q) = (i', q')$, and terminate the trajectory with probability $1 - \gamma$.

Alternatively, we could perform "batch" updates, waiting until trajectories terminate before making adjustments to estimates and controller parameters.

This account is necessarily terse; implementational details and empirical results will be presented at the workshop. We remark that this approach should apply to POMDP's as well, and that, in contrast to approach presented in (Duff, 2000), the prior need not be a member of a conjugate family of distributions (e.g., Dirichlet). Current research seeks more robust Monte-Carlo estimation techniques.

References

Duff, M. O. (2000) "Reinforcement learning for Bayes-adaptive Markov decision processes," Research Note 3/25/00, http://envy.cs.umass.edu/ People/duff/duff.html.

Meuleau, N., Kim, K. E., Kaelbling, L. P., & Cassandra, A. R. (1999) "Solving POMDPs by searching the space of finite policies," UAI-99.